①

HANDY REFERENCE CARD

valFORTH 1.1 T.M.

Stack inputs and outputs are shown; top of stack on right.
This card follows usage of the Forth Interest Group (S.F. Bay Area); usage aligned with the Forth 78 International Standard.

For more info: Forth Interest Group
P.O. Box 1105
San Carlos, CA 94070.

Operand Key:		
	n,n1,...	16-bit signed numbers
	d,d1,...	32-bit signed numbers
	u	16-bit unsigned number
	addr	address
	b	8-bit byte
	c	7-bit ascii character value
	f	boolean flag
	fp	floating point number
	$	string

Stack Manipulation

DUP	(n -- n n)	Duplicate top of stack.
DROP	(n --)	Throw away top of stack.
SWAP	(n1 n2 -- n2 n1)	Reverse top two stack items.
OVER	(n1 n2 -- n1 n2 n1)	Make copy of second item on top.
ROT	(n1 n2 n3 -- n2 n3 n1)	Rotate third item to top.
<ROT	(n1 n2 n3 -- n3 n1 n2)	Rotate top item to third.
-DUP	(n -- n ?)	Duplicate only if non-zero.
>R	(n --)	Move top item to "return stack" for temporary storage (use caution).
R>	(-- n)	Retrieve item from return stack.
R	(-- n)	Copy top of return stack onto stack.

Number Bases

DECIMAL	(--)	Set decimal base.
HEX	(--)	Set hexadecimal base.
BASE	(-- addr)	System variable containing number base.

Arithmetic and Logical

+	(n1 n2 -- sum)	Add.
D+	(d1 d2 -- sum)	Add double-precision numbers.
-	(n1 n2 -- diff)	Subtract (n1-n2).
*	(n1 n2 -- prod)	Multiply.
/	(n1 n2 -- quot)	Divide (n1/n2).
MOD	(n1 n2 -- rem)	Modulo (i.e. remainder from division).
/MOD	(n1 n2 -- rem quot)	Divide, giving remainder and quotient.
*/MOD	(n1 n2 n3 -- rem quot)	Multiply, then divide (n1*n2/n3), with double-precision intermediate.
*/	(n1 n2 n3 -- quot)	Like */MOD, but give quotient only.
MAX	(n1 n2 -- max)	Maximum.
MIN	(n1 n2 -- min)	Minimum.
ABS	(n -- absolute)	Absolute value.
DABS	(d -- absolute)	Absolute value of double-precision number.
MINUS	(n -- -n)	Change sign.
DMINUS	(d -- -d)	Change sign of double-precision number.
AND	(n1 n2 -- and)	Logical AND (bitwise).
OR	(n1 n2 -- or)	Logical OR (bitwise).
XOR	(n1 n2 -- xor)	Logical exclusive OR (bitwise).
NOT	(n -- f)	True if top number zero (i.e. reverses truth value).

Comparison

<	(n1 n2 -- f)	True if n1 less than n2.
>	(n1 n2 -- f)	True if n1 greater than n2.
<=	(n1 n2 -- f)	True if n1 less than or equal to n2.
>=	(n1 n2 -- f)	True if n1 greater than or equal to n2.
=	(n1 n2 -- f)	True if top two numbers are equal.
<>	(n1 n2 -- f)	True if n1 does not equal n2.
0<	(n -- f)	True if top number negative.
0>	(n -- f)	True if top number positive.
0=	(n -- f)	True if top number zero (i.e. reverses truth value.
0#	(n -- f)	True if n does not equal zero.

Memory

@	(addr -- n)	Replace word address by contents.
!	(n addr --)	Store second word at address on top.
C@	(addr -- b)	Fetch one byte only.
C!	(b addr --)	Store one byte only.
?	(addr --)	Print contents of address.
C?	(addr --)	Print byte at address.
U?	(addr --)	Print unsigned contents of address.
+!	(n addr --)	Add second number on stack to contents of address on top.
CMOVE	(from to u --)	Move u bytes in memory from head to head.
<CMOVE	(from to u --)	Move u bytes in memory from tail to tail.
FILL	(addr u b --)	Fill u bytes in memory with b, beginning at address.
ERASE	(addr u --)	Fill u bytes in memory with zeroes, beginning at address.
BLANKS	(addr u --)	Fill u bytes in memory with blanks, beginning at address.

Control Structures

DO...LOOP	do: (end+1 start --)	Set up loop, given index range.
I	(-- index)	Place current index value on stack.
I'	(-- index)	Used to retrieve index after a >R.
J	(-- index)	Place index of outer DO-LOOP on stack.
LEAVE	(--)	Terminate loop at next LOOP, +LOOP, or /LOOP.
?EXIT	(--)	LEAVE if ?TERMINAL is true (i.e. pressed).
DO... +LOOP	do: (end+1 start --) +loop: (n --)	Like DO...LOOP, but adds stack value (instead of always '1') to index.
DO... /LOOP	do: (end+1 start --) /loop: (u --)	Like DO... +LOOP, but adds unsigned value to index.
IF...(true) ...ENDIF	if: (f --)	If top of stack true (non-zero), execute. (Note: Forth 78 uses IF...THEN.)
IF...(true) ...ELSE ...(false) ...ENDIF	if: (f --)	Same, but if false, execute ELSE clause. (Note: Forth 78 uses IF...ELSE...THEN.)
BEGIN... UNTIL	until: (f --)	Loop back to BEGIN until true at UNTIL. (Note: Forth 78 uses BEGIN...END.)
BEGIN... WHILE ...REPEAT	while: (f --)	Loop while true at WHILE;REPEAT loops unconditionally to BEGIN. (Note: Forth 78 uses BEGIN...IF ...AGAIN.)

Terminal Input – Output

.	(n --)	Print number.
.R	(n fieldwidth --)	Print number, right-justified in field.
D.	(d --)	Print double-precision number
D.R	(d fieldwidth --)	Print double-precision number, right-justified in field.
CR	(--)	Do a carriage return.
SPACE	(--)	Type one space.
SPACES	(n --)	Type n spaces.
."	(--)	Print message (terminated by ").
DUMP	(addr u --)	Dump u words starting at address.
TYPE	(addr u --)	Type string of u characters starting at address.
COUNT	(addr -- addr+1 u)	Change length-byte string to TYPE form.
?TERMINAL	(-- f)	True if terminal break request present.
KEY	(-- c)	Read key, put ascii value on stack.
EMIT	(c --)	Type ascii value from stack.
EXPECT	(addr n --)	Read n characters (or until carriage return) from input to address.
WORD	(c --)	Read one word from input stream, using given character (usually blank) as delimiter.

Input – Output Formating

NUMBER	(addr -- d)	Convert string at address to double-precision number.
<#	(--)	Start output string.
#	(d -- d)	Convert next digit of double-precision number and add character to output string.
#S	(d -- 0 0)	Convert all significant digits of double-precision number to output string.
SIGN	(n d -- d)	Insert sign of n into output string.
#>	(d -- addr u)	Terminate output string (ready for TYPE).
HOLD	(c --)	Insert ascii character into output string.

Disk Handling

LIST	(screen --)	List a disk screen.
LOAD	(screen --)	Load disk screen (compile or execute).
BLOCK	(block -- addr)	Read disk block to memory address.
B/BUF	(-- n)	System constant giving disk block size in bytes.
BLK	(-- addr)	System variable containing current block number.
SCR	(-- addr)	System variable containing current screen number.
UPDATE	(--)	Mark last buffer accessed as updated.
FLUSH	(--)	Write all updated buffers to disk.
EMPTY-BUFFERS	(--)	Erase all buffers.

Defining Words

: xxx	(--)	Begin colon definition of xxx.
;	(--)	End colon definition.
VARIABLE xxx	(n --) xxx: (-- addr)	Create a variable named xxx with initial value n; returns address when executed.
CONSTANT xxx	(n --) xxx: (-- n)	Create a constant named xxx with value n; returns value when executed.
CODE xxx	(--)	Begin definition of assembly-language primitive operative named xxx.
;CODE	(--)	Used to create a new defining word, with execution-time "code routine" for this data type in assembly.
<BUILDS... DOES>	does: (-- addr)	Used to create a new defining word, with execution-time routine for this data type in higher-level Forth.
LABEL xxx	(-- addr)	Creates a header xxx which when executed returns its PFA.

HANDY REFERENCE CARD

valFORTH 1.1 T.M.

valFORTH 6502 Assembler

ASSEMBLER	(---)	Calls up the assembler vocabulary for subsequent assembly language programming.
CODE xxx	(---)	Enters the new word "xxx" into the dictionary as machine language word and calls up the assembler vocabulary for subsequent assembly language programming.
C;	(---)	Terminates an assembly language definition by performing a security check and setting the CONTEXT vocabulary to the same as the CURRENT vocabulary.
END-CODE	(---)	A commonly used synonym for the word C; above. The word C; is recommended over END-CODE.
SUBROUTINE	xxx (---)	Enters the new word "xxx" into the dictionary as machine language subroutine and calls up the assembler vocabulary for subsequent assembly language programming.
;CODE	(---)	When the assembler is loaded, puts the system into the assembler vocabulary for subsequent assembly language programming. See main glossary for further explanation.

Control Structures

IF,	(flag --- addr 2)	Begins a machine language control structure based on the 6502 status flag on top of the stack. Leaves an address and a security check value for the ELSE, or ENDIF, clauses below. "flag" can be EQ , NE , CC , CS , VC , VS , MI , or PL . Command forms: ...flag..IF,..if-true..ENDIF,...all... ...flag..IF,..if-true.. ELSE,..if-false..ENDIF,..all...
ELSE,	(addr 2 --- addr 3)	Used in an IF, clause to allow for execution of code only if IF, clause is false. If the IF, clause is true, this code is bypassed.
ENDIF,	(addr 2/3 ---)	Used to terminate an IF, control structure clause. Additionally, ENDIF, resolves all forward references. See IF, above for command form.
BEGIN,	(--- addr 1)	Begins machine language control structures of the following forms: ...BEGIN,...AGAIN,... ...BEGIN,...flag..UNTIL,... ...BEGIN,...flag..WHILE,..while-true..REPEAT,... where "flag" is one of the 6502 statuses: EQ , NE , CC , CS , VC , VS , MI , and PL .
UNTIL,	(addr 1 flag ---)	Used to terminate a post-testing BEGIN, clause thus allowing for conditional looping of a program segment while "flag" is false.
WHILE,	(addr 1 flag --- addr 4)	Used to begin a pre-testing BEGIN, clause thus allowing for conditional looping of a program segment while "flag" is true.
REPEAT,	(addr 4 ---)	Used to terminate a pre-testing BEGIN,..WHILE, clause. Additionally, REPEAT, resolves all forward addresses of the current WHILE, clause.
AGAIN,	(addr 1 ---)	Used to terminate an unconditional BEGIN, clause. Execution cannot exit this loop unless a JMP, instruction is used.

Parameter Passing (These routines must be jumped to.)

NEXT	(--- addr)	Transfers control to the next FORTH word to be executed. The parameter stack is left unchanged.
PUSH	(--- addr)	Pushes a 16 bit value to the parameter stack whose low byte is found on the 6502 return stack and whose high byte is found in the accumulator.
PUSHOA	(--- addr)	Pushes a 16 bit value to the parameter stack whose low byte is found in the accumulator and whose high byte is zero.
PUT	(--- addr)	Replaces the value currently on top of the parameter stack with the 16 bit value whose low byte is found on the 6502 stack and whose high byte is in the accumulator.
PUTOA	(--- addr)	Replaces the value currently on top of the parameter stack with the 16 bit value whose low byte is in the accumulator and whose high byte is set to zero.
BINARY	(--- addr)	Drops the top value of the parameter stack and then performs a PUT operation described above.
POP and POPTWO	(--- addr)	POP drops one value from the parameter stack. POPTWO drops two values from the parameter stack.
SETUP	(--- addr)	Moves one to four values to the N scratch area in the zero page and drops all values moved from the parameter stack.
N	(--- addr)	Points to a nine-byte scratch area in the zero page beginning at N-1 and going to N+7.
Opcodes	(various --- various)	ADC, AND, ASL, BIT, BRK, CLC, CLD, CLI, CLV, CMP, CPX, CPY, DEC, DEX, DEY, EOR, INC, INX, INY, JSR, JMP, LDA, LDX, LDY, LSR, NOP, ORA, PHA, PHP, PLA, PLP, ROL, ROR, RTI, RTS, SBC, SEC, SED, SEI, STA, STX, TAX, TAY, TSX, TXA, TXS, TYA,

Aliases

```
NXT,  = NEXT JMP,        POP2, = POPTWO JMP,
PSH,  = PUSH JMP,        XL,   = XSAVE LDX,
PUT,  = PUT JMP,         XS,   = XSAVE STX,
PSHA, = PUSHOA JMP,      THEN, = ENDIF,
PUTA, = PUTOA JMP,       END,  = UNTIL,
POP,  = POP JMP,
```

HANDY REFERENCE CARD

valFORTH 1.1 T.M.

Graphics and Color

SETCOLOR	(n1 n2 n3 --)	Color register n1 (0...3 and 4 for background) is set to hue n2 (0 to 15) and luminance n3 (0-14, even).
SE.	(n1 n2 n3 --)	Alias for SETCOLOR.
GR.	(n --)	Identical to GR. in BASIC. Adding 16 will suppress split display. Adding 32 will suppress display preclear. In addition, this GR. will not disturb player/missiles.
POS.	(x y --)	Same as BASIC POSITION or POS. Positions the invisible cursor if in a split display mode, and the text cursor if in 0 GR. .
POSIT	(x y --)	Positions and updates the cursor, similar t PLOT, but without changing display data.
PLOT	(x y --)	Same as BASIC PLOT. PLOTs point of color register specified by last COLOR command, point x y.
DRAWTO	(x y --)	Same as BASIC DRAWTO. Draws line from las PLOT'ted, DRAWTO'ed or POSIT'ed point to x using color in register specified by last C LOR command.
DR.	(x y --)	Alias for DRAWTO.
FIL	(b --)	Fills area between last PLOT'ted, DRAWTO'ed or POSIT'ed point to last position set by POS., using the color in register b.
G"	(--)	Used in the form G" ccccc". Sends text cccc to text area in non-0 Graphics mode, starting at current cursor position, in color of register specified by last COLOR command prior to cccc being output.
GTYPE	(addr count --)	Starting at addr, output count characters to text area in non-0 Graphics mode, starting at current cursor position, in color of register specified by last COLOR command.
LOC.	(x y -- b)	Positions the cursor at x y and fetches the data from display at that position. Like BASIC LOCATE and LOC. .
(G")	(--)	Run-time code compiled in by G".
POS@	(-- x y)	Leaves the x and y coordinates of the cursor on the stack.
CPUT	(b --)	Outputs the data b to the current cursor position.
CGET	(-- b)	Fetches the data b from the current cursor position.
>SCD	(c1 -- c2)	Converts c1 from ATASCII to its display screen code, c2. Example: ASCII A >SCD 88 @ C! will put an "A" into the upper left corner of the display.
SCD>	(c1 -- c2)	Converts c1 from display screen code to ATASCII c2. See >SCD.
>BSCD	(addr1 addr2 count --)	Moves count bytes from addr1 to addr2, translating from ATASCII to display screen code on the way.
BSCD>	(addr1 addr2 count --)	Moves count bytes from addr1 to addr2, translating from display screen code to ATASCII on the way.
COLOR	(b --)	Saves the value b in the variable CLRBYT.
CLRBYT	(-- addr)	Variable that holds data from last COLOR command.

GREY	-- 0	PINK	-- 4	BLUE	-- 8	GREEN	-- 12
GOLD	-- 1	LVNDR	-- 5	LTBLUE	-- 9	YLWGRN	-- 13
ORNG	-- 2	BLPRPL	-- 6	TURQ	-- 10	ORNGRN	-- 14
RDORNG	-- 3	PRPLBL	-- 7	GRNBL	-- 11	LTORNG	-- 15

(CONSTANTs)

SOUND	(chan freq dist vol --)	Sets up the sound channel "chan" as indicated. Channel: 0-3 Frequence: 0-255, 0 is highest pitch. Distortion: 0-14, evens only. Volume: 0-15. Suggested mnemonic: CatFish Don't Vote
SO.	(chan freq dist vol --)	Alias of SOUND.
FILTER!	(n --)	Stores n in the audio control register and into the valFORTH shadow register, AUDCTL. Use AUDCTL when doing bit manipulation, then do FILTER!.
AUDCTL	(-- addr)	A variable containing the last value sent to the audio control register by FILTER!.
XSND	(n --)	Silences channel n.
XSND4	(--)	Silences all channels.

Text Output and Disk Preparation

S:	(flag --)	If flag is true, enables handler that sends text to text screen. If false, disables the handler. (See PFLAG in main glossary.)
P:	(flag --)	If flag is true, enables handler that sends text to printer. If false, disables the handler. (See PFLAG in main glossary)
BEEP	(--)	Makes a raucous noise from the keyboard.
ASCII	(c, -- n (executing)) (c, -- (compiling))	Converts next character in input stream to ATASCII code. If executing, leaves on stack. If compiling, compiles as literal.
EJECT	(--)	Causes a form feed on smart printers if the printer handler has been enabled by ON P:. May need adjustment for dumb or nonstandard printers.
LISTS	(start count --)	From start, lists count screens. May be aborted by CONSOLE button at the end of a screen.
PLIST	(scr --)	Lists screen scr to the printer, then restores former printer handler status.
PLISTS	(start cnt --)	From start, lists cnt screens to printer three to a page, then restores former printer handler status. May be aborted by CONSOLE button at the end of a screen.
FORMAT	(--)	With prompts, will format a disk in drive of your choice.

Debugging Utilities

DECOMP	xxx	Does a decompilation of the word xxx if it can be found in the active vocabularies.
CDUMP	(addr n --)	A character dump from addr for at least n characters. (Will always do a multiple of 16.)
#DUMP	(addr n --)	A numerical dump in the current base for at least n characters. (Will always do a multiple of 8.)
(FREE)	(-- n)	Leaves number of bytes between bottom of display list and PAD.
FREE	(--)	Does (FREE) and then prints the stack and "bytes".
H.	(n --)	Prints n in HEX, leaves BASE unchanged.
STACK	(flag --)	If flag is true, turns on visible stack. If flag is false, turns off visible stack.
.S	(... -- ...)	Does a signed, nondestructive stack printout, TOS at right. Also sets visible stack to do signed printout.
U.S	(... -- ...)	Does unsigned, nondestructive stack printout, TOS at right. Also sets visible stack to do unsigned printout.
B?	(--)	Prints the current base, in decimal. Leaves BASE undisturbed.
CFALIT	xxx (-- cfa (executing)) xxx (-- (compiling))	Gets the cfa (code field address) of xxx. If executing, leaves it on the stack; if compiling, compiles it as a literal.

Floating Point

FCONSTANT	xxx (fp --) xxx (-- fp)	The character string is assigned the constant value fp. When xxx is executed, fp will be put on the stack.
FVARIABLE	xxx (fp --) xxx: (addr --)	The character string xxx is assigned the initial value fp. When xxx is executed, the addr (two bytes) of the value of xxx will be put on the stack.
FDUP	(fp1 -- fp1 fp1)	Copies the fp number at top-of-stack.
FDROP	(fp --)	Discards the fp number at top-of-stack.)
FOVER	(fp2 fp1 -- fp2 fp1 fp2)	Copies the fp number at 2nd-on-stack to top-of-stack.
FLOATING	xxx (-- fp)	Attempts to convert the following string, xxx, to a fp number.
FP	xxx (-- fp)	Alias for FLOATING.
F@	(addr -- fp)	Fetches the fp number whose address is at top-of-stack.
F!	(fp addr --)	Stores fp into addr. Remember that the operation will take six bytes in memory.
F.	(fp --)	Type out the fp number at top-of-stack. Ignores the current value in BASE and uses base 10.
F?	(addr --)	Fetches a fp number from addr and types it out.
F+	(fp2 fp1 -- fp3)	Replaces the two top-of-stack fp items, fp2 and fp1, with their fp sum, fp3.
F-	(fp2 fp1 -- fp3)	Replaces the two top-of-stack fp items fp2 and fp1, with their difference, fp3=fp2-fp1.
F*	(fp2 fp1 -- fp3)	Replaces the two top-of-stack fp items fp2 and fp1, with their product, fp3.
F/	(fp2 fp1 -- fp3)	Replaces the two top-of-stack fp items fp2 and fp1, with their quotient, fp3=fp2/fp1.
FLOAT	(n -- fp)	Replaces number at top-of-stack with its fp equivalent.
FIX	(fp (non-neg, less than 32767.5) -- n)	Replaces fp number at top-of-stack, constrained as indicated, with its integer equivalent.
LOG	(fp1 -- fp2)	Replaces fp1 with its base e logarithm, fp2. Not defined for fp1 negative.
LOG10	(fp1 -- fp2)	Replaces fp1 with its base 10 decimal logarithm, fp2. Not defined for fp1 negative.
EXP	(fp1 -- fp2)	Replaces fp1 with fp2, which equals e to the power fp1.
EXP10	(fp1 -- fp2)	Replaces fp1 with fp2, which equals 10 to the power fp1.
F0=	(fp -- flag)	If fp is equal to floating-point 0, a true flag is left. Otherwise, a false flag is left.
F=	(fp2 fp1 -- flag)	If fp2 is equal to fp1, a true flag is left. Otherwise, a false flag is left.
F>	(fp2 fp1 -- flag)	If fp2 is greater than fp1, a true flag is left. Otherwise, a false flag is left.
F<	(fp2 fp1 -- flag)	If fp2 is less than fp1, a true flag is left. Otherwise, a false flag is left.
FLITERAL	(fp --)	If compiling, then compile the fp stack value as a fp literal.

Operating System

OPEN	(addr n0 n1 n2 -- n3)	This word opens the device whose name is at addr. The device is opened on channel n0 with AUX1 and AUX2 as n1 and n2 respectively. The device status byte is returned as n3.
CLOSE	(n --)	Closes channel n.
PUT	(b1 n -- b2)	Outputs byte b1 on channel n, returns status byte b2.
GET	(n -- b1 b2)	Gets byte b1 from channel n, returns status byte b2.
GETREC	(addr n1 n2 -- n3)	Inputs record from channel n2 up to length n1. Returns status byte n3.
PUTREC	(addr n1 n2 -- n3)	Outputs n1 characters starting at addr through channel n2. Returns status byte n3.
STATUS	(n -- b)	Returns status byte b from channel n.
DEVSTAT	(n -- b1 b2 b3)	From channel n1 gets device status bytes b1 and b2, and normal status byte b3.
SPECIAL	(b1 b2 b3 b4 b5 b6 b7 b8 -- b9)	Implements the Operating System "Special" command. AUX1 through AUX6 are b1 through b6 respectively, command byte is b7, channel number is b8. Returns status byte b9.
RS232	(--)	Loads the Atari 850 drivers into the dictionary (approx 1.8K).

HANDY REFERENCE CARD

valFORTH 1.1 T.M.

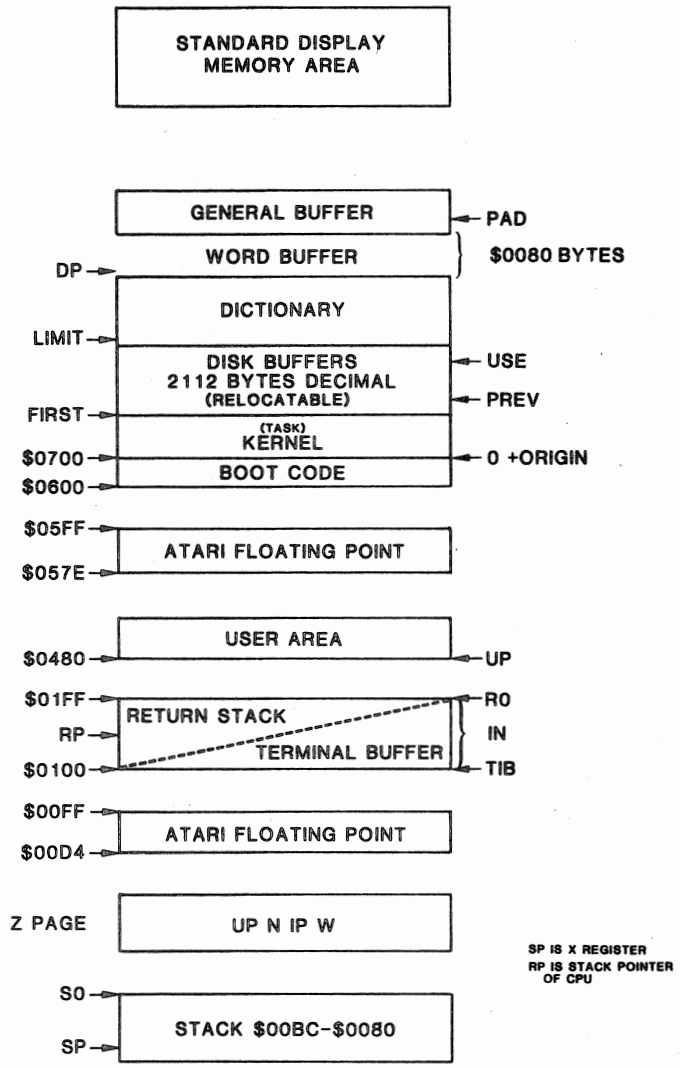
Vocabularies

CONTEXT	(-- addr)	Returns address of pointer to context vocabulary (searched first).
CURRENT	(-- addr)	Returns address of pointer to current vocabulary (where new definitions are put).
FORTH	(--)	Main Forth vocabulary (execution of FORTH sets CONTEXT vocabulary).
EDITOR	(--)	Editor vocabulary; sets CONTEXT.
ASSEMBLER	(--)	Assembler vocabulary; sets CONTEXT.
DEFINITIONS	(--)	Sets CURRENT vocabulary to CONTEXT.
VOCABULARY xxx	(--)	Create new vocabulary named xxx.
VLIST	(--)	Print names of all words in CONTEXT vocabulary.

Miscellaneous and System

(	(--)	Begin comment, terminated by right paren on same line; space after (.
FORGET xxx	(--)	Forget all definitions back to and including xxx.
ABORT	(--)	Error termination of operation.
'xxx	(-- addr)	Find the address of xxx in the dictionary; if used in definition, compile address.
HERE	(-- addr)	Returns address of next unused byte in the dictionary.
PAD	(-- addr)	Returns address of scratch area (usually 128 bytes beyond HERE).
IN	(-- addr)	System variable containing offset into input buffer. Used, e.g., by WORD.
SP@	(-- addr)	Returns address of top stack item.
ALLOT	(n --)	Leave a gap of n bytes in the dictionary.
,	(n --)	Compile a number into the dictionary.

valFORTH T.M. Memory Map



HANDY REFERENCE CARD

valFORTH™

SOFTWARE SYSTEM

EDITOR 1.1 COMMAND SUMMARY

Below is a quick reference list of all the commands which the video editor recognizes.

Entering the Edit Mode (executed outside of the edit mode)

V	(scr# ---)	* Enter the edit mode and view the specified screen
L	(---)	* Re-view the current screen.
WHERE	(---)	* Enter the edit mode and position the cursor over the word that caused a compilation error.
LOCATE cccc	(---)	Enter the edit mode and position the cursor o the word "cccc" where it is defined.
LOCATOR	(ON/OFF ---)	When ON, allows all words compiled until the OFF to be locatable using the LOCATE command .
#BUFS	(#lines --)	Sets the length (in lines) of the storage buffe The default is five.

Cursor Movement (issued within the edit mode)

ctrl	↑	* Move cursor up one line, wrapping to the bottom line if moved off the top.
ctrl	↓	* Move cursor down one line, wrapping to the top line if moved off the bottom.
ctrl	←	* Move cursor left one character, wrapping to the right edge if moved off the left.
ctrl	→	* Move cursor right one character, wrapping to the left edge if moved off the right.
RETURN		Position the cursor at the beginning of the next line.
TAB		Advance to next tabular column.

Editing Commands (issued within the edit mode)

ctrl	INS	Insert one blank at cursor location, losing the last character on the line.
ctrl	DEL	Delete character under cursor, closing the line.
shift	INS	* Insert blank line above current line, losing the last line on the screen.
shift	DEL	* Delete current cursor line, closing the screen.
ctrl	I	Toggle insert-mode/replace-mode. (see full description of ctrl-I).
BACKS		* Delete last character typed, if on the same line as the cursor.
ctrl	H	Erase to end of line (Hack).

Buffer Management (issued within the edit mode)

ctrl	T	Delete current cursor line sending it to the edit buffer for later use.
ctrl	F	Take the current buffer line and insert it above the current cursor line.
ctrl	K	Copy current cursor line sending it to the edit buffer for later use.
ctrl	U	Take the current* buffer line and copy it to the current cursor line.
ctrl	R	Roll the buffer making the topmost buffer line current.
ctrl	B	Roll the buffer backwards making the fourth buffer line on the screen current.
ctrl	C	Clear the current* buffer line and performs a ctrl-B.

*Note: The current buffer line is bottommost on the video display.

Changing Screens (issued within the edit mode)

ctrl	P	Display the previous screen saving all changes made to the current screen.
ctrl	N	Display the next screen saving all changes made to the current screen.
ctrl	S	* Save the changes made to the current screen and end the edit session.
ctrl	Q	* Quit the edit session forgetting all changes made to the current screen.

Special Keys (issued within the edit mode)

ESC		* Do not interpret the next key typed as any of the commands above. Send it directly to the screen instead.
ctrl	A	Put the arrow "-->" ("next screen") in the lower-right-hand corner of the screen unless it is already there, in which case remove it.
ctrl	J	Split the current line into two lines at the point where the cursor is.
ctrl	O	Corrects any major editing blunders.

Screen Management (executed outside of the edit mode)

FLUSH	(--)	* Save any updated FORTH screens to disk.
EMPTY-BUFFERS	(--)	* Forget any changes made to any screens not yet FLUSHed to disk.
COPY	(from to --)	* Copies screen #from to screen #to.
CLEAR	(scr# --)	* Blank fills specified screen.
CLEARS	(scr# #screens --)	Blank fills the specified number of screens starting with screen scr#.
SMOVE	(from to #screens --)	Duplicate the specified number of screens Starting with screen number "from".

* EDITOR 1.0 COMMAND

HANDY REFERENCE CARD

valFORTH T.M.

SOFTWARE SYSTEM

GENERAL UTILITIES

Strings

UMOVE	(addr1 addr2 n --)	UMOVE is a "universal" memory move. It takes the block of memory n bytes long at addr1 and copies it to memory location addr2. UMOVE correctly uses either CMOVE or <CMOVE.
" ccc"	(--)	(at compile time)
	(-- addr)	(at run time) If compiling, the sequence ccc (delimited by the trailing ") is compiled into the dictionary as a string: | len | c | c | c |...| c |
$CONSTANT	xxx ($ --	(at compile time)
	xxx: (-- $)	(at execution time) Takes the string on top of the stack and compiles it into the dictionary with the name xxx. When xxx is later executed, the address of the string is pushed onto the stack.
$VARIABLE	xxx (n --)	Reserves space for a string of length n.
	xxx: (-- $)	When xxx is later executed, the address of the string is pushed onto the stack.
$.	($ --)	Takes the string on top of the stack and sends it to the current output device.
$!	($ addr --)	Takes the string at second on stack and stores it at the address on top of stack.
$+	($1 $2 -- $3)	Takes $2 and concatenates it with $1, leaving $3 at PAD.
LEFT$	($1 n -- $2)	Returns the leftmost "n" characters of $1 as $2.
RIGHT$	($1 n -- $2)	Returns the rightmost "n" characters of $1 as $2.
MID$	($1 n u -- $2)	Returns $2 of length u starting with the nth character of $1.
LEN	($ -- len)	Returns the length of the specified string.
ASC	($ -- c)	Returns the ASCII value of the first character of the specified string.
$COMPARE	($1 $2 -- flag)	Compares $1 with $2 and returns a status flag.
$=	($1 $2 -- flag)	Compares two strings on top of the stack.
$<	($1 $2 -- flag)	Compares two strings on top of the stack.
$>	($1 $2 -- flag)	Compares two strings on top of the stack.
SAVE$	($1 -- $2)	As most string operations leave resultant strings at PAD, the word SAVE$ is used to temporarily move strings to PAD+512.
INSTR	($1 $2 -- n)	Searches $1 for first occurrence of $2. Returns the character position in $1 if a match is found; otherwise, zero is returned.
CHR$	(c -- $)	Takes the character "c" and makes it into a string of length one and stores it at PAD.
DSTR$	(d -- $)	Takes the double number d and converts it to its ASCII representation as $ at PAD.
STR$	(n -- $)	Takes the single length number n and converts it to its ASCII representation as $ at PAD.
STRING$	(n $1 -- $2)	Creates $2 as n copies of the first character of $1.
#IN$	(n -- $)	#IN$ has three similar but different functions. If n is positive, it accepts a string of n or fewer characters from the terminal. If n is zero, it accepts up to 255 characters from the terminal. If n is negative, it returns only after accepting -n characters from the terminal. The resultant string is stored at PAD.
IN$	(-- $)	Accepts a string of up to 255 characters from the terminal.
$-TB	($1 -- $2)	Removes trailing blanks from $1 leaving new $2.
$XCHG	($1 -- $2)	Exchanges the contents of $1 with $2.

Array Word Glossary

ARRAY	xxx (n --)	(compiling)
	xxx:(m -- addr)	(executing) When compiling, creates an array named xxx with n 16-bit elements numbered 0 thru n-1. Initial values are undefined. When executing, takes an argument, m, off the stack and leaves the address of element m of the array.
CARRAY	xxx (n --)	(compiling)
	xxx:(m -- addr)	(executing) When compiling, creates a c-array named xxx with n 8-bit elements numbered 0 thru n-1. Initial values are undefined. When executing, takes an argument, m, off the stack and leaves the address of element m of the c-array.
TABLE	xxx (--)	(compiling)
	xxx:(m -- addr)	(executing) When compiling, creates a table named xxx but does not allot space. Elements are compiled in directly with , (comma). When executing, takes one argument, m off the stack and, assuming 16-bit elements, leaves the address of element m of the table.
CTABLE	xxx (--)	(compiling)
	xxx:(m -- addr)	(executing) When compiling, creates a c-table named xxx but does not allot space. Elements are compiled in directly with C, (c-comma). When executing, takes one argument, m off the stack and, assuming 8-bit elements, leaves the address of element m of the c-table.
VECTOR	xxx (n0 ... nN count ..)	(compiling)
	xxx:(m -- addr)	(executing) When compiling, creates a vector named xxx with count 16-bit elements numbered 0-N. n0 is the initial value of element 0, nN is the initial value of element N, and so on. When executing, takes one argument, m, off the stack and leaves the address of element m on the stack.
CVECTOR	xxx (b0 ... bN count --)	(compiling)
	xxx:(m -- addr)	(executing) When compiling, creates a c-vector named xxx with count 8-bit elements numbered 0-N. b0 is the initial value of element 0, bN is the initial value of element N, and so on. When executing, takes an argument, m, off the stack and leaves the address of element m on the stack.

Double Number Extensions

DVARIABLE	xxx (d --)	At compile time, creates a double number variable xxx with the initial value d. At run time, xxx leaves the address of its value on the stack.
	xxx:(-- addr)	
DCONSTANT	xxx (d --)	At compile time, creates a double number constant xxx with the initial value d. At run time, xxx leaves the value d on the stack.
	xxx:(-- d)	
D-	(d1 d2 -- d3)	Leaves d1-d2=d3.
D0=	(d -- flag)	If d is equal to 0. leaves true flag; otherwise, leaves false flag.
D=	(d1 d2 -- flag)	If d1 equals d2, leaves true flag; otherwise, leaves false flag.
D0<	(d -- flag)	If d is negative, leaves true flag; otherwise, leaves false flag.
D<	(d1 d2 -- flag)	If d1 is less than d2, leaves true flag; otherwise, leaves false flag.
D>	(d1 d2 -- flag)	If d1 is greater than d2, leaves true flag; otherwise, leaves false flag.
DMIN	(d1 d2 -- d3)	Leaves the minimum of d1 and d2.
DMAX	(d1 d2 -- d3)	Leaves the maximum of d1 and d2.
D>R	(d --)	Sends the double number at top of stack to the return stack.
DR>	(-- d)	Pulls the double number at top of the return stack to the stack.
D,	(d --)	Compiles the double number at top of stack into the dictionary.
DU<	(ud1 ud2 -- flag)	If the unsigned double number ud1 is less than the unsigned double number ud2, leaves a true flag; otherwise, leaves a false flag.
M+	(d1 n -- d2)	Converts n to a double number and then sums with d1.

High Resolution Text Output

GCINIT	(--)	Initializes the graphic character output routines. This must be executed prior to using any other hi-res output words.
GC.	(n --)	Displays the single length number n at the current hi-res cursor location.
GC.R	(n1 n2 --)	Displays the single length number n1 right-justified in a field n2 graphic characters wide. See .R .
GCD.R	(d n --)	Displays the double length number d right-justified in a field n graphic characters wide. See D.R .
GCEMIT	(c --)	Displays the text character c at the current hi-res cursor location. Three special characters are interpreted by GCEMIT.
GCLEN	(addr n -- len)	Scans the first n characters at addr and returns the number of characters that will actually be displayed on screen.
GCR	(--)	Repositions the hi-res cursor to the beginning of the next hi-res text line. See CR .
GCLS	(--)	Clears the hi-res display and repositions the cursor in the upper lefthand corner.
GCSPACE	(--)	Sends a space to the graphic character output routine. See SPACE .
GCSPACES	(n --)	Sends n spaces to the graphic character output routine. See SPACES .
GCTYPE	(addr n --)	Sends the first n characters at addr to the graphic character output routine. See TYPE .
GC" ccc"	(--)	Sends the character string ccc (delimited by ") to the graphic character output routine.
GCBKS	(--)	Moves the hi-res cursor back one character position for overstriking or underlining.
GCPOS	(horz vert --)	Positions the hi-res cursor to the coordinates specified. Note that the upper lefthand corner is 0,0.
GC$.	(addr --)	Sends the string found at addr and preceded by a count byte to the graphic character output routine. See $. .
SUPER	(--)	Forces the graphic character output routine into the superscript mode (or out of the subscript mode). See VMI below. May be performed within a string by the ^ character.
SUB	(--)	Forces the graphic character output routine into the subscript mode (or out of the superscript mode). See VMI below. May be performed within a string by the v character.
VMI	(n --)	The VMI command sets the number of eighths of characters to scroll up or down when either a SUPER or SUB command is issued.
VMI#	(-- addr)	A variable set by VMI.
OSTRIKE	(ON or OFF --)	If the OSTRIKE option is ON, characters are printed over top of the previous characters giving the impression of overstriking.
GCBAS	(-- addr)	A variable which contains the address of the character set displayed by GCEMIT. To change character sets, simply store the address of your new character set into this variable.
GCLFT	(-- addr)	A variable which holds the column position of the left margin.
GCRGT	(-- addr)	A variable which holds the column position of the right margin.

Case Structures

CASE: structure

Format:

```
CASE:  wordname
       word0
       word1
       ...
       wordN  ;
```

CASE Structure

Format:

```
: wordname
  ...
  CASE
     word0
     word1
     ...
     wordN
( NOCASE  wordnone )
  CASEND
  ... ;
```

SEL Structure

Format:

```
: wordname
  ...
  SEL
     n1  ->  word0
     n2  ->  word1
     ...
     nN   >  wordN
(  NOSEL  wordnone  )
  SELEND
  ... ;
```

COND Structure

Format:

```
: wordname
  ...
  COND
     condition0  <<  words0  >>
     condition1  <<  words1  >>
     ...
     conditionN  <<  wordsn  >>
(  NOCOND  wordsnone  )
  CONDEND
  ... ;
```

Miscellaneous Utilities

XR/W	(#secs addr blk flag --)	"Extended read-write." The same as R/W except that XR/W accepts a sector count for multiple sector reads and writes. Starting at address addr and block blk, read (flag true) or write (flag false) #secs sectors from or to disk.
LOADS	(start count --)	Loads count screens starting from screen # start.
THRU	(start finish -- start count)	Converts two range numbers to a start-count format.
SEC	(n --)	Provides an n second delay. Uses a tuned do-loop.
MSEC	(n --)	Provides an n millisecond delay. (approx) Uses a tuned do-loop.
H->L	(n1 -- n2)	Moves the high byte of n1 to the low byte and zero's the high byte, creating n2. Machine code.
L->H	(n1 -- n2)	Moves the low byte of n1 to the high byte and zero's the low byte, creating n2. Machine code.
H/L	(n1 -- n1(hi) n1(lo))	Split top of stack into two stack items: New top of stack is low byte of old top of stack. New second on stack is old top of stack with low byte zeroed.
BIT	(b -- n)	Creates a number n that has only its bth bit set. The bits are numbered 0-15.
?BIT	(n b -- f)	Leaves a true flag if the bth bit of n is set. Otherwise leaves a false flag.
TBIT	(n1 b -- n2)	Toggles the bth bit of n1, making n2.
SBIT	(n1 b -- n2)	Sets the bth bit of n1, making n2.
RBIT	(n1 b -- n2)	Resets the bth bit of n1, making n2.
STICK	(n -- horz vert)	Reads the nth stick (0-3) and resolves the setting into horizontal and vertical parts, with values from -1 to +1. -1 -1 means up and to the left.
PADDLE	(n1 -- n2)	Reads the n1th paddle (0-7) and returns its value n2. Machine code.
16TIME	(-- n)	Returns a 16 bit timer reading from the system clock at locations 19 and 20, decimal.
8RND	(-- b)	Leaves one random byte from the internal hardware. Machine code.
16RND	(-- n)	Leaves one random word from the internal hardware. Machine code with 20 cycle extra delay for rerandomization.
CHOOSE	(u1 -- u2)	Randomly choose an unsigned number u2 which is less than u1.
CSHUFL	(addr n --)	Randomly rearrange n bytes in memory, starting at address addr.
SHUFL	(addr n --)	Randomly rearrange n words in memory, starting at address addr.
DUMP	(addr n --)	Starting at addr, dump at least n bytes (even multiple of 8) as ASCII and hex. May be exited early by pressing a CONSOLE button.
BXOR	(addr count b --)	Starting at address addr, for count bytes, perform bit-wise exclusive OR with byte b at each address.
BAND	(addr count b --)	Starting at address addr, for count bytes, perform bit-wise AND with byte b at each address.
BOR	(addr count b -)	Starting at address addr, for count bytes, perform bit-wise OR with byte b at each address.
STRIG	(n -- flag)	Reads the button of joystick n (0-3).
PTRIG	(n -- flag)	Reads the button of paddle n (0-7).

HANDY REFERENCE CARD

valFORTH™ SOFTWARE SYSTEM

PLAYER-MISSILE GRAPHICS, CHARACTER EDITOR, & SOUND EDITOR

Player/Missile Command Summary

Note: Players and missiles are numbered 0 thru 3. The fifth player is numbered 4.

(PMINIT)	(addr res ---)	Initializes the player missile routines with PM memory specified by "addr" with "res" resolution.
PMINIT	(res ---)	Initializes the player missile routines with "res" resolution and with PM memory located at the first available memory below the display list.
PMBAS	(--- addr)	A variable pointing to player/missile memory which is set by (PMINIT) or PMINIT. It can be read from but not written to.
PLAYERS	(ON/OFF ---)	This command enables or disables the player/missile graphic display.
5THPLY	(ON/OFF ---)	This command turns (the fifth player mode) ON or OFF. If OFF, missiles take the colors of their corresponding players. If ON, all missiles take on the common color of playfield 3. The fifth player is numbered as four (4).
PLYCLR	(pl# ---)	Erases the specified player (0-3,4).
MSLCLR	(ml# ---)	Erases the specified missile (0-3).
PMCLR	(---)	Erases all players and all missiles.
MCPLY	(ON/OFF ---)	This command turns (the multiple color player mode) ON or OFF. See documentation for explanation.
PRIOR	(n ---)	Sets the priority of players and playfields. See documentation for legal settings.
PLYWID	(width pl# ---)	Sets the width of the specified player. Legal widths are normal (0 or 2), double (1), or quadruple (3).
MSLWID	(width ml# ---)	Sets the width of the specified missile. Legal widths are normal (0 or 2), double (1), or quadruple (3).
PMCOL	(pl# hue lum ---)	Sets the specified player to the color defined by "hue" and "lum".
BLDPLY	(addr len horz vert pl# ---)	Creates a player whose image is at "addr" with a length "len". The player is originally placed at the specified horizontal and vertical coordinates.
BLDMSL	(addr len horz vert ml# ---)	Creates a missile whose image is at "addr" with a length "len". The player is originally placed at the specified horizontal and vertical coordinates.
PLYLOC	(pl# --- horz vert)	Returns the horizontal and vertical coordinates of the specified player.
MSLLOC	(ml# --- horz vert)	Returns the horizontal and vertical coordinates of the specified missile.
PLYMV	(horz vert pl# ---)	Moves the specified player according to the horizontal and vertical offsets specified. A positive horizontal offset moves the player right, a negative one moves it left. Likewise, a positive vertical offset moves the player down and a negative one moves it up.
MSLMV	(horz vert ml# ---)	Moves the specified missile according to the horizontal and vertical offsets specified. See PLYMV above.
PLYPUT	(x y pl# ---)	Positions the specified player and location (x,y) on the video display.
PLYCHG	(addr len pl# ---)	This changes the image of the specified player to the image of length "len" at "addr".
PLYSEL	(addr # pl# ---)	This changes the image of the specified player to image number "#" in a table of images starting at address "addr".
PLYBND	(l r t b pl# ---)	Specified the left, right, top, and bottom boundaries of the specified player.
MSLBND	(l r t b ml# ---)	Specified the left, right, top, and bottom boundaries of the specified missile.
?BND	(--- n)	Returns the boundary status of the last player or missile moved. See documentation for a description of this value.
?PLYSTT	(pl# --- n)	Returns the boundary status of the last move of the specified player. See documentation for a description of this value.
?MSLSTT	(ml# --- n)	Returns the boundary status of the last move of the specified missile. See documentation for a description of this value.
?COL	(--- f)	Returns true (1) if any collisions have occurred since the last HITCLR command was issued.
?MXPF	(ml# --- n)	Returns 0 if the specified missile has not hit any playfields since the last HITCLR command. If any collisions have occurred, a status value is returned. See documentation.
?PXPF	(pl# --- n)	Returns 0 if the specified player has not hit any playfields since the last HITCLR command. If any collisions have occurred, a status value is returned. See documentation.
?MXPL	(ml# --- n)	Returns 0 if the specified missile has not hit any players since the last HITCLR command. If any collisions have occurred, a status value is returned. See documentation.
?PXPL	(pl# --- n)	Returns 0 if the specified player has not hit any other players since the last HITCLR command. If any collisions have occurred, a status value is returned.
HITCLR	(---)	Clears the collision registers to a no-collision state.

Audio Editor Command Summary

AUDED	(--)	Calls up the audio-palette program.

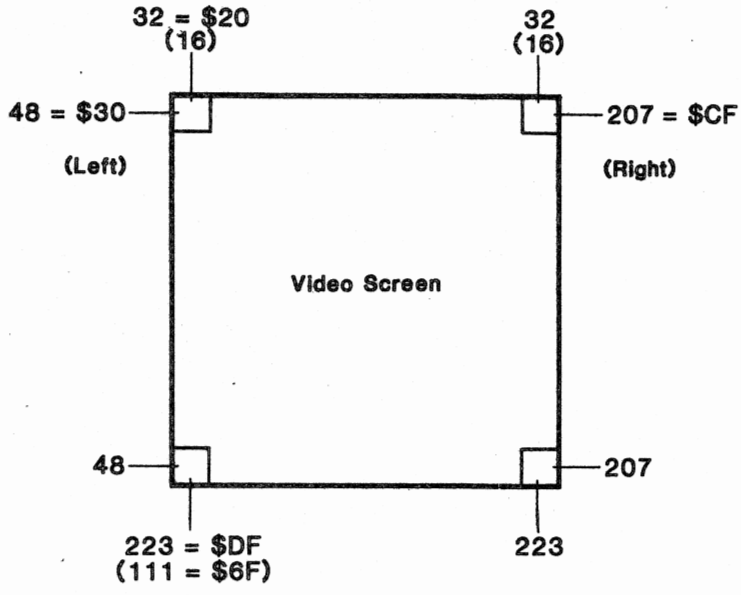
Character Editor Command Summary

CHAR-EDIT	(--)	Calls up the character editor.

PLAYER-MISSILE BOUNDARY MAP

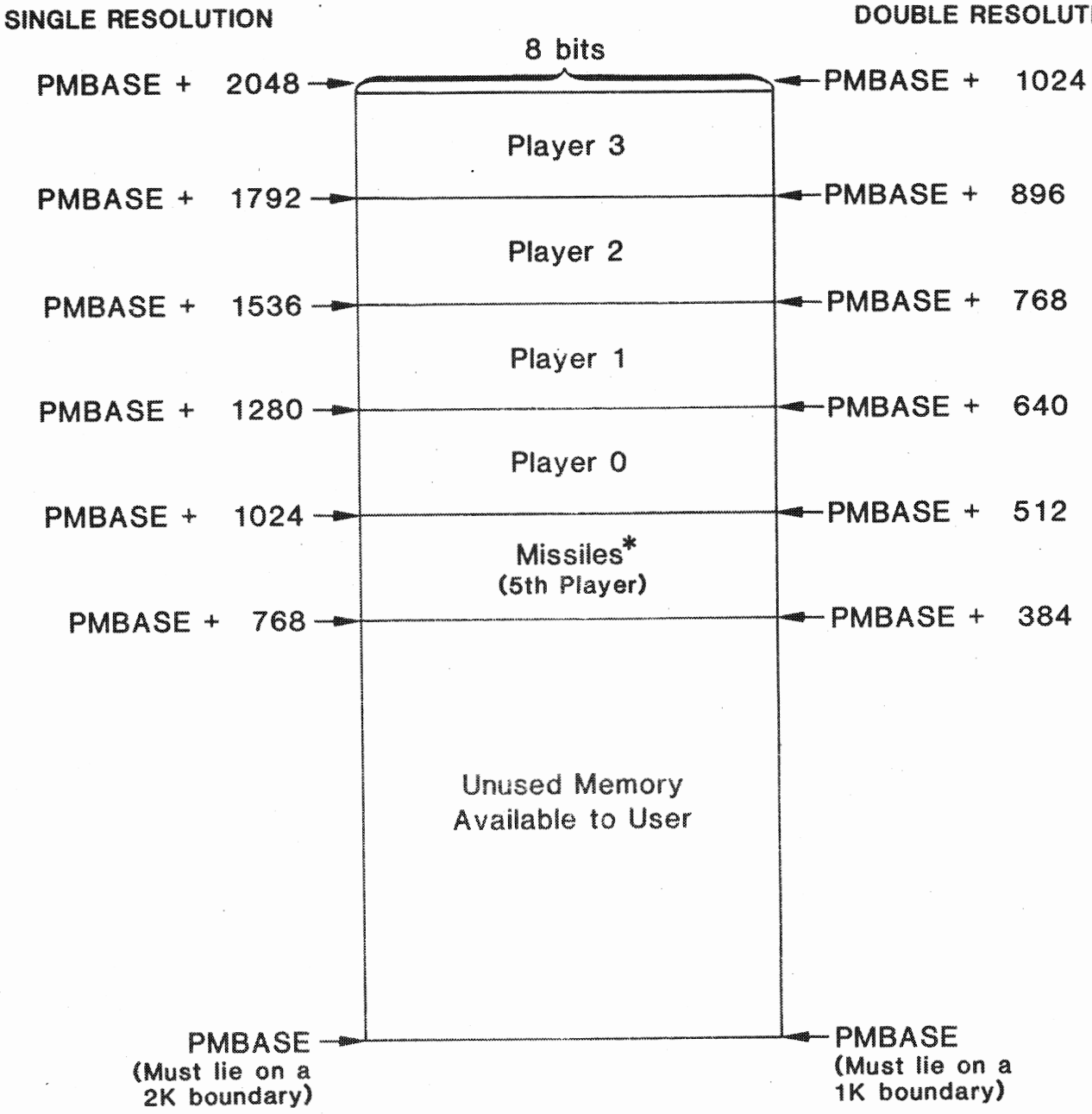
valFORTH

(Double resolution values are in parentheses)



PLAYER-MISSILE Memory Map

valFORTH™



*Note: All missiles occupy the same memory location. This is possible because unlike players which are 8 bits wide and fill an entire byte, missiles are only two bits wide. Four missiles can therefore be represented in the same amount of memory as a single player.

Byte form: ! m3 ! m2 ! m1 ! m0 !

HANDY REFERENCE CARD

valFORTH T.M.

SOFTWARE SYSTEM

DISPLAY FORMATTER

(DBINIT)	(dmem dlist ---)	This command initializes the display formatter using "dlist" as the target address for the display list, and "dmem" as the target address for display memory.
DBINIT	(---)	This command initializes the display formatter setting the display memory address to top-of-memory minus $1F00. The display list is targeted for $100 bytes below the display memory.
DBM	(opcode ---)	The DBM command adds "opcode" to the end of the current display list.
DBMS	(#times opcode ---)	The DBMS command performs a multiple DBM command as described above.
DBPTR	(block# ---)	This command makes the specified block the next block to be created with the DBM comma It essentially makes block#-1 the end of the current list.
DBIN	(opcode block# ---)	The DBIN command inserts the specified opcode into the specified block. That block and all following blocks are pushed back one block.
DBINS	(#times opcode block# ---)	This command performs a multiple DBIN command as described above.
DBDEL	(block # ---)	The DBDEL command deletes the specified block from the current display list.
DBDELS	(#times block# ---)	This command performs a multiple DBDEL command as described above.
DBDELL	(---)	The DBDELL command deletes the last block of the current disolay list.
DBMOD	(modifier block# ---)	This command modifies the specified block. Legal modifiers are VRTMOD, HRZMOD, and INTMOD.
DBMODL	(modifier ---)	This command modifies the last block of the current display list.
DBREM	(block# ---)	The DBREM command removes all modifiers from the specified block. Note that if a HRZMOD is stripped, display memory allocation will change.
DBREMS	(#times block# ---)	This command is a multiple DBREM command.
DBREML	(---)	This command strips modifiers from the last block of the current display list.
?DBVAL	(block# --- value)	The ?DBVAL command returns all information about the display block specified, i.e., the antic mode and any modifiers. This information is returned as one value.
?ANTMOD	(block# --- antic-mode)	This command returns the antic-mode (or opcode) of the specified block.
?DBMODS	(block# --- modifiers)	The ?DBMODS command returns the modifiers for the specified block. This information is returned as one value. See documentation for notes on interpretation of this value.
DBWID	(width ---)	The DBWID command sets the display formatter up for narrow (1), normal (2), or wide (3) screen display.
DBADR	(block# --- address)	Given a display block number, it returns the address of the first byte of that display block.
DMCLR	(---)	The DMCLR command clears the display memory pointed to by the display list currently being created.
USRDSP	(---)	Once a display list has been created, USRDSP activates the new list.
MIXED	(---)	The MIXED command performs a USRDSP then instructs the Atari operating system to re-direct all output to the video display memory specified by the newly created display list.
DMPLST	(---)	The DMPLST command gives a complete, informative listing of the display list currently being created.
DSPEND	(--- address)	A variable which contains a pointer to the end of the current display list. It is an offset from 0 DSPLST.
DSPBLK	(--- address)	A variable containing the number of the next display block to be created.
DMLOC	(--- address)	A variable which contains the target address of the display memory pointed to by the current display list.
LSTLOC	(--- address)	A variable which contains the target address for the current display list.
DBLST	(block# --- address)	An array of addresses used by DBADR.
DSPLST	(pointer --- address)	A c-array containing the display list currently being created. DSPEND above points to the end of the list in this array.

HANDY REFERENCE CARD

valFORTH T.M.

SOFTWARE SYSTEM

DISPLAY FORMATTER

The Character Modes

There are 6 character modes (opcodes 2 thru 7). All character modes work in the same way, i.e., the values in display memory are indices to a large "n" by 8 byte array. In some of these modes, the highest one or two bits are used to specify a color with only the remaining lower bits used for indexing. The following table gives information about each of the modes:

Antic mode	2	3	4	5	6	7
Basic mode	0	---	---	---	1	2
# color *	1.5	1.5	5	5	5	5
Chars/line narrow wid	32	32	32	32	16	16
Chars/line normal wid	40	40	40	40	20	20
Chars/line wide screen	48	48	48	48	24	24
Scan lines/ pixel	8	10	8	16	8	16
Bits/pixel	1	1	2	2	1	1
Color clocks per pixel	.5	.5	1	1	1	1

Colors

mode 2: Takes the color of PF2 with the lum of PF1
(Artifacting/bleed very noticeable)
mode 3: Same as above
mode 4: Two bits/pixel in character definitions
00 = BAK 01 = PF0 10 = PF1
11 = PF2 if bit 7 of index = 0, else PF3
mode 5: Same as 4 above
mode 6: Most significant two bits of index
0 = PF0 1 = PF1 etc.
mode 7: Same as 6 above

The Graphic Modes

There are 8 graphic modes. Unlike character modes, the values in display memory are not indices into an array of character definitions, but rather are the definitions themselves. Depending on the graphic mode, these values give different results. The following table gives various information about each mode.

Antic mode	8	9	A	B	C	D	E	F*
Basic mode	3	4	5	6	---	7	---	8
# colors	4	2	4	2	2	4	4	1.5
bytes/line narrow wid	8	8	16	16	16	32	32	32
bytes/line normal wid	10	10	20	20	20	40	40	40
bytes/line wide screen	12	12	24	24	24	48	48	48
Pixels per normal wid	40	80	80	160	160	160	160	320
Scan lines/pixel	8	4	4	2	1	2	1	1
Bits/pixel	2	1	2	1	1	2	2	1
Color clocks per pixel	4	2	2	1	1	1	1	.5

*Mode F values differ when in GTIA modes

Colors

mode 8: Two bits/pixel, 4 pixels/byte
00 = BAK 01 = PF0 10 = PF1 11 = PF2
mode 9: One bit/pixel, 8 pixels/byte
0 = BAK 1 = PF0
mode A: Same as mode 8 above
mode B: Same as mode 9 above
mode C: Same as mode 9 above
mode D: Same as mode 8 above
mode E: Same as mode 8 above
mode F: Take the color of PF2 and lum of PF1
(if not in a GTIA mode)

HANDY REFERENCE CARD

valFORTH T.M.

SOFTWARE SYSTEM

valGRAPHICS T.M.

General Functions

PEN	(n --)	This command is used to change the color which the armadillo draws and fills.
PHPEN	(n --)	This command is used to change the color which the armadillo fills.
DRAW	(n --)	Move the armadillo n units in the directi in which it is heading. Draw that portior the line of travel of the armadillo that f within the current window.
DRAWTO	(x y --)	Move the armadillo to x y and draw that po of the line of travel that falls within thc current window.
PHIL	(n --)	Move the armadillo n spaces in the directio it is heading, and color that portion of thc path of travel with the PEN value. Also perform a fill to the right during the time that the armadillo is in the current window.
PHILTO	(x y --)	Move the armadillo to the point x, y. Then proceed as in PHIL.
GO	(n --)	GO moves the armadillo n units in the directi n in which it is facing.
DUPGO	(n -- n)	Same as GO, but doesn't destroy stack argument.
GO.	(n --)	Same as GO, but colors last pixel with PEN color.
DUPGO.	(n -- n)	Same as GO., but doesn't destroy stack argument.
DOT	(--)	DOT puts a pen-color dot at the present armadillo position.
GOTO	(x y --)	GOTO positions the armadillo at x,y.
GOTO.	(x y --)	Same as GOTO but puts dot at x,y.
CENTER	(--)	Positions the armadillo at the point 0,0.
CENTERO	(--)	Positions the armadillo at the point 0,0 and turns it to face up.
RELOC	(--)	Positions the armadillo at the last point drawn by the system routines. DILLO vocab.
ASPECT	(ON or OFF --)	ON ASPECT will cause vertical components of subsequent graphics commands to be scaled to account for pixels not being square.
DX1	(-- n)	Returns the x coordinate of the armadillo.
DY1	(-- n)	Returns the y coordinate of the armadillo.
TURN	(n --)	Changes the direction that the armadillo is facing by n degrees clockwise.
TURNTO	(n --)	Turns the armadillo to a heading of n degrees clockwise from vertical.
TURNTWD	(x y --)	Turns the armadillo so that it faces toward the point x,y.
DAZM	(-- n)	Returns the direction, in degrees (0-359), in which the armadillo is facing.

Windows and Coordinate Systems

WINDOW	(left right top bottom --)	Sets a new window whose boundaries, expressed in the coordinate system of the base window (not the current window), are taken from the stack in the order indicated.
RELWND	(left right top bottom --)	Makes current a window whose edges are as indicated on stack in the coordinate system of the current window (not the base window).
WIPE	(--)	Colors the entire current window according to the color register selected by the last PHBAK command, but uses DRAW and draw options.
FRAME	(--)	Draws a line around the current window.
BASWND	(--)	Makes the base window (usually the full window first put up by a GR. command) current, centers the armadillo and turns it to 0 degrees.
THISWND	xxx, (--) xxx: (--)	Creates a word, xxx, which when executed makes current the window which was current at the time xxx was defined.
DEFBAS	(left right top bottom --)	Advanced users. Used to set up a base window when not using GR.. The values indicated are the number of pixels from the left edge of the display (for left and right) and from the top edge of the display (for top and bottom). DILLO.
WCTR	(--)	Center the armadillo in the current window.
WCTRO	(--)	Center the armadillo in the current window and turn it to 0 degrees.

Line-naming/Line Manipulation and Point-naming

NAMEPT	xxx, (x y --) xxx: (-- x y)	Creates a word xxx. When xxx is executed, it returns x and y to the stack.
THISPT	xxx, (----) xxx: (-- x y)	Creates a word xxx. xxx returns defining-time armadillo x,y.
2PT-LN	(x1 y1 x2 y2 -- a b c)	Takes the coordinates of two points and leaves a, b, and c of the connecting line.
MAKLN	(-- a b c)	Pushes to stack the a, b, c representation of the imaginary line along which the armadillo faces.
NAMELN	xxx, (a b c --) xxx: (-- a b c)	Creates the word xxx. When xxx is executed, it returns the values a b c to the stack.
THISLN	xxx, (--) xxx: (-- a b c)	Creates the word xxx. When xxx is executed, it returns the a, b, and c values of the line that the armadillo was sitting on and facing along when xxx was created.
2LNX	(a1 b1 c1 a2 b2 c2 -- x y)	Given two lines on the stack in a b c form, 2LNX returns the point of intersection of the two lines.

Options

(All words below take a flag stack argument, and leave none.)

Switch	Default	ON	OFF
RPHIL	on	Enables right fill with PHIL, PHILTO	Disables right fill with PHIL, PHILTO.
LPHIL	off	Enables left fill with PHIL, PHILTO.	Disables left fill with PHIL, PHILTO.
DRXOR	off	DRAW, DRAWTO will xor pixels with line color.	DRAW, DRAWTO will replace pxls with line color.
PHXOR	off	PHIL, PHILTO will xor pixels with fill color.	PHIL, PHILTO will replace pxls with fill color.
DRUNT	off	Enable draw-until functions.	Disable draw-until functions.
PHUNT	off	Fill to edge of window or to dest. pixel.	Fill until encountering halt pixel cond set by PHBAK, PHUNOT.
DRUNOT	on	With DRUNT on, DRAW, DRAWTO draw until hit color set by DRBAK, PHBAK.	With DRUNT on, DRAW, DRAWTO draw until hit not color set by DRBAK, PHBAK.
PHUNOT	on	With PHUNT on, PHIL, PHILTO fill until hitting color set by PHBAK.	With PHUNT on, PHIL, PHILTO fill until hitting not color set by PHBAK.
PH+DR	on	PHIL, PHILTO draw line as filling.	PHIL, PHILTO don't draw line as filling.
DR1ST	on	First point of lines is drawn.	First point of lines is not drawn.
PHCRNR	off	PHIL, PHILTO perform corner checking, armadillo must be moving vertically.	No corner checking.

DINIT sets all switches to their default values.

HANDY REFERENCE CARD
valFORTH™ SOFTWARE SYSTEM
Text Compression and Auto Text Formatting

Basic Commands

*."	(--)	Sends following string of characters to the formatter.
*TYPE	(addr count --)	Sends count characters starting at addr to the formater.
*CR	(--)	*CR formats and flushes the buffer to the output device, clears the buffer, does CR.
*EMIT	(c --)	Sends the character c to the formatter.
*SPACE	(--)	Sends a single character of value in the quan BKGND to the formatter, through *EMIT.
*SPACES	(n --)	Sends n characters of value in the quan BKGND to the formatter, through *EMIT.
*BACKS	(--)	Backs up the formatter buffer pointer, BPTR, one location and fills new location with BKGND value.
RGTJST	(--)	Sets up formatter for right justification.
LFTJST	(--)	Sets up formatter for left justification.
CTRJST	(--)	Sets up formatter for center justification.
FILJST	(--)	Sets up formatter for fill justification.
INVID	(f --)	ON INVID means text will be output in inverse video; OFF INVID means normal video.
INVBK	(f --)	ON INVBK means background of text will be output in inverse video. OFF INVID means normal video.
CAP	(--)	Causes capitalization of the next byte processed by *EMIT or *TYPE.
CAPS	(f --)	ON CAPS means text will be capitalized if lower case. OFF CAPS means text will be printed as-is.
COLOR	(b --)	Color register b will be used for color of subsequent text output to windows in Graphics modes 1 and 2.
TYPEOUT	(--)	TYPEOUT directs the formatter to use TYPE as its actual output routine, allowing output to the display screen or printer.
WINDOUT	(--)	WINDOUT directs the formatter to use window routines for output. A window must be created before attempting to use window output.
INVBK	(ON or OFF --)	When ON, background character output by formatter in 0 graphics mode will be inverse video blank. When OFF, this character will be normal video blank.

Text Compression

W=	xxx, (--) xxx: (--)	Creates a tc-word-compiling word, named xxx, and a headerless tc-word which when executed sends the string xxx through the formatter followed by *SPACE. xxx when executed, compiles in the cfa of this tc-word. W= and xxx are both in transient area and so are disposed by DISPOSE.
P=	xxx, (--) xxx: (--)	Creates a tc-prefix-compiling word, named xxx, and a headerless tc-prefix which when executed sends the string xxx through the formatter. xxx, when executed, compiles in the cfa of this cfa of this tc-prefix. P= and xxx are both in the transient area and so are disposed by DISPOSE.
S=	xxx, (--) xxx: (--)	Creates a tc-suffix-compiling word, named xxx and a headerless tc-suffix which when executed sends the string xxx through the formatter preceded by *BACKS and followed by *SPACE. xxx, when executed, compiles in the cfa of this tc-suffix. S= and xxx are both in the transient area and so are disposed by DISPOSE.

Typed Output

PRTWID	(-- n)	A quan containing the width of the area to be printed when printer output from the formatter has been selected by PRT:.
PRTIND	(-- n)	A quan containing the number of spaces the printer is to indent when outputting from the formatter.
PWID	(-- n)	A quan containing the number of columns the printer is actually able to print.
VIDIND	(-- n)	A quan containing the number of spaces the output routine is to indent when outputting from the formatter.
VIDWID	(-- n)	A quan containing the width of the area to be written when video output from the formatter has been selected by VID:.
PRT:	(--)	Directs TYPEd output to the printer, and moves appropriate values into WWID and PVIND.
VID:	(--)	Directs TYPEd output to the video display, and moves appropriate values into WWID and PVIND.
PRINIT	(--)	Resets PCTR, the printed line counter.

Windows

WADR	(--)	Address in memory corresponding to character position in upper lefthand corner of current window.
WHGT	(--)	Height in lines of currently active window.
WCLR	(--)	Fills the current window with BKGND.
NAMWND	(wadr wid hght b/ch byt/ln --)	One of many possible window-defining structures. Accepts window upper lefthand corner address, its width, height, byte/character, and the bytes/ln of the current graphics mode.
NAMEBW	xxx, (column row wid hgt --) xxx: (--)	Names a 0 graphics window for later activation.
MAKEBW	(col row wid hgt --)	Establishes a 0 graphics window immediately but does not name it for later retrieval.
NAMECW	xxx, (col row wid hgt --) xxx: (--)	Names a 1 or 2 graphics window for later activation.
MAKECW	(col row wid hgt --)	Establishes a 0 graphics window immediately but does not name it for later retrieval.

Virtual (Disk-based) Memory

(A pointer to a byte on disk is implemented by the two system variables, BLK and IN in the fig model. BLK contains the block number pointed to and IN contains the number of bytes into the block the byte in question is located.)

V"	(-- blk in)	Leaves the values of BLK and IN on the stack at the time it is executed and then scans the virtual memory pointer formed by BLK and IN forward until the next " character is encountered.
XMTV	(--)	Starting from the location in virtual memory pointed to by BLK and IN, outputs characters through *EMIT until a " character is encountered, which it does not output.
XCOUNT	(adr -- adr+2 xcount)	Extracts a two-byte count from an extended string, and leaves the count on top of the address + 2.
M:	xxx, (blk in --) xxx: (--)	Generally used after V". Takes a virtual memory pointer from the stack, and creates a word xxx which when executed will push the virtual memory pointer to BLK and IN and then execute XMTV, thus retrieving a message from disk.
V:	xxx, (blk in --) xxx: (--)	Creates a word xxx which when executed pushes the virtual memory pointer which was on stack at the time of its creation to BLK and IN.
V$TP	(-- XCOUNT)	Extracts a two-byte string count from the disk location to which BLK and IN point, leaves it on stack, and bumps the virtual memory pointer made up of BLK and IN twice.
V$@	(-- X$=PAD)	Extracts the extended string in virtual memory pointed to by BLK and IN. The string is left at PAD.
V$*EMT	(--)	Sends the extended string pointed to by BLK and IN through *EMIT.
V$!	(X$ --)	Stores the extended string on stack to virtual memory starting at the location pointed to by BLK and IN.
X"	(---x$-PAD)	Reads the following characters until the delimeter " as an extended string and stores the string at PAD. Operates from screens only. Crosses block and screen boundaries without additional code.
ALTINIT	(scr --)	Sets up ALTBLK and ALTIN to point to screen scr. ALTBLK and ALTIN form an auxiliary virtual memory pointer that is used to keep track of how far messages have been compiled onto the destination disk.